Report

Incremental Clustering: Intrusion Detection by Video Surveillance

Name: Keya Shukla

- **Problem Statement**

  Implement a basic incremental K Means algorithm using the iris dataset (available in scikit-learn module or can download the csv file). Once done, try using the same algorithm for intrusion detection using any video of your choice (Only one such video is required as our algorithm will learn as the new frames are introduced automatically).

- **Prerequisites**

  - Software:
    - Python 3 (Use anaconda as your python distributor as well)

  - Tools:
    - Numpy
    - Pandas
    - Seaborn
    - Matplotlib
    - Sklearn

  - Dataset: Iris dataset from Sklearn and A video for Clustering

- **Method Used**

  The incremental clustering approach is a way to address the dynamically growing dataset. It attempts to minimize the scanning and calculation effort for newly added data points. It is essential to efficiently store and utilize knowledge about the existing clustering results for incremental clustering.

  BIRCH (balanced iterative reducing and clustering using hierarchies) is an unsupervised data mining algorithm used to perform hierarchical clustering over particularly large data-sets.

  Detection of abnormalities in live videos requires optimized scene representation which involves real-time detection of objects while efficiently representing the state of objects temporally across frames. For such purposes, Incremental Clustering can be used.

Incremental clustering allows clustering of pixels with motion which is further used for mapping the trajectories in subsequent frames and can be used for Surveillance and for real-time traffic analysis.

- **Implementation:**

1. Load all required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import Birch
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import os
import sys
import cv2
```

2. Calling Iris dataset from SKLearn

```
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(data= np.c_[iris['data']],columns= iris['feature_names'])
```

```
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

3. Implementing Birch clustering

```
clf = Birch(threshold=0.5, branching_factor=50, n_clusters=3, compute_labels=True, copy=True)
```

```
clf.fit(df)
```

```
Birch()
```

```
pred = clf.predict(df)
```

```
pred
```

4. Implementing Function for initializing background

```
def initBackground(initImage):
    img_arr = mpimg.imread(initImage)
    mean = img_arr
    variance = 9*np.ones(img_arr.shape)
    return(mean,variance)
```

5. Implementing Function for initializing fore background

```
def ForegroundDetection(img_file,mean,variance,lmda):
    img = mpimg.imread(img_file)
    d = img - mean
    y = variance*(lmda**2)
    d_2 = np.square(d)
    I = d_2 - y
    mask = np.all(I>0,axis=2)
    rI = 255*mask.astype(int)
    rI = rI.astype(np.uint8)
    return(rI)
```

6. Implementing Function for background subtraction

```
def Background_Subtraction(img_dir,lmda,eta,m,n,alpha):

    img_file_name = os.listdir(img_dir)
    initImage = os.path.join(img_dir,img_file_name[0])
    mean, variance = initBackground(initImage)

    for i in range(1,len(img_file_name)):
        img_path = os.path.join(img_dir,img_file_name[i])

        fig, ax = plt.subplots(1,3,figsize=(10,10))
        rI = ForegroundDetection(img_path,mean,variance,lmda)
        ax[0].imshow(rI,cmap="gray")

        cI = Voting(rI,eta,m,n)
        mean, variance = meanvarUpdate(cI,img_path,mean,variance,alpha)
        ax[1].imshow(cI,cmap="gray")

        img = mpimg.imread(img_path)
        ax[2].imshow(img,cmap="gray")

        plt.show()
```

- **Results:**

1. Resulting images from Incremental Clustering: